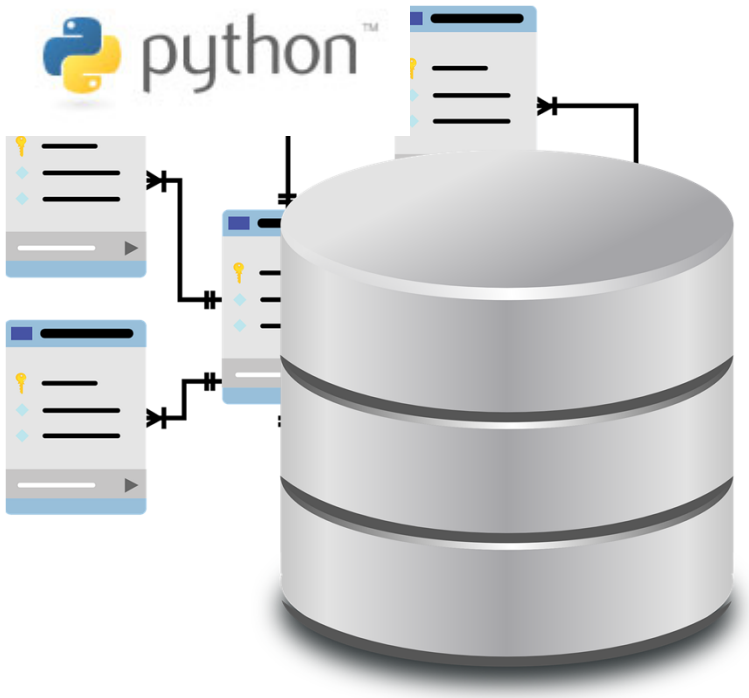
Error Handling in Python

Error Handling

- try - except
  - Goes to except if the try condition fails (exception occurs)
  - Programmer decides what to do if try fails
  - Can have multiple except blocks based on error types:
    - OSError, ValueError, RuntimeError, TypeError, NameError
    - Or user-defined errors
- try - except - else
  - else block will be processed if no exception
- Try – except – finally
  - finally will be processed regardless
    - Good for Clean ups

```
def isValidDigit(s):
    try:
        # If s is a valid digit, then no error from casting
        result = float(s)
        print("it is a number")
        return True
    except ValueError:
        print("it is not a number")
        return False
```

Python and MySQL Integration - Revisited



Python and MySQL Integration – Data Cleaning

```
import pandas as pd
import numpy as np

df = pd.read_csv('cookies dirty data.csv')

# drop rows if all elements are blank
df.dropna(how="all", inplace = True)

# remove leading/trailing spaces
df["Salesman"] = df["Salesman"].str.strip()

# replaces invalid values with NaN for one column
df['Price'] = df['Price'].replace('[^A-Za-z0-9]',np.NaN,regex=True)

# replaces invalid values with NaN for multiple columns
df[['Tweets','Sales']] = df[['Tweets','Sales']].replace('[^0-9]',np.NaN,regex=True)

# drop rows if one or more elements are blank
df = df.dropna()
```

Python and MySQL Integration – DateTime Object

	Date	Day	Temperature	Salesman	Tweets	Price	Sales
0	1/1/2019	Tuesday	72.0	John	2	0.5	177
2	1/3/2019	Thursday	69.0	John	5	0.5	172
3	1/4/2019	Friday	100.0	John	7	0.5	150
5	1/6/2019	Sunday	91.0	Ada	8	0.5	120
6	1/7/2019	Monday	81.0	Ada	3	0.3	96

```
df['Date'] = pd.to_datetime(df['Date'], format = '%m/%d/%Y')
```

	Date	Day	Temperature	Salesman	Tweets	Price	Sales
0	2019-01-01	Tuesday	72.0	John	2	0.5	177
2	2019-01-03	Thursday	69.0	John	5	0.5	172
3	2019-01-04	Friday	100.0	John	7	0.5	150
5	2019-01-06	Sunday	91.0	Ada	8	0.5	120
6	2019-01-07	Monday	81.0	Ada	3	0.3	96

Python and MySQL Integration – DataTime Object

	Date	Day	Temperature	Salesman	Tweets	Price	Sales
0	1/1/2019	Tuesday	72.0	John	2	0.5	177
2	1/3/2019	Thursday	69.0	John	5	0.5	172
3	1/4/2019	Friday	100.0	John	7	0.5	150
5	1/6/2019	Sunday	91.0	Ada	8	0.5	120
6	1/7/2019	Monday	81.0	Ada	3	0.3	96

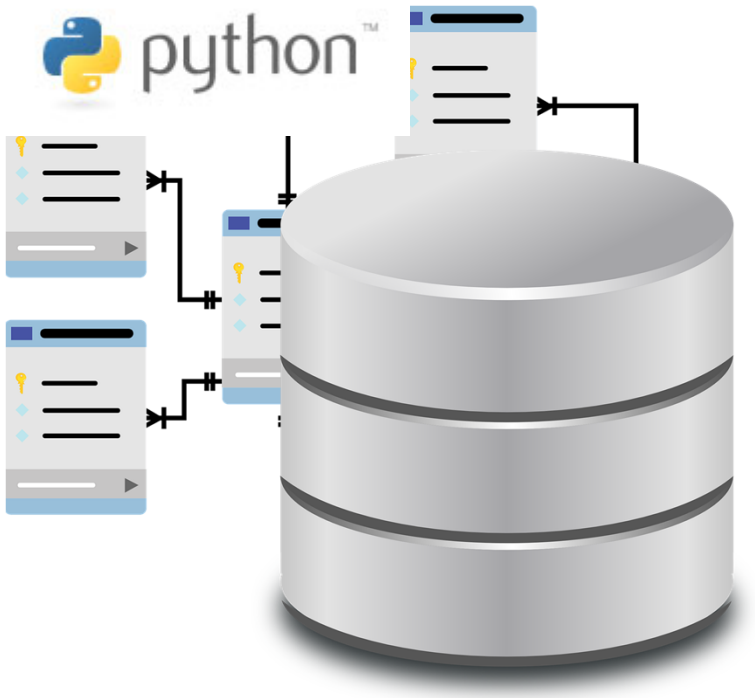
```
df = pd.read_csv('cookies dirty data.csv', parse_dates=['Date'])
```

	Date	Day	Temperature	Salesman	Tweets	Price	Sales
0	2019-01-01	Tuesday	72.0	John	2	0.5	177
2	2019-01-03	Thursday	69.0	John	5	0.5	172
3	2019-01-04	Friday	100.0	John	7	0.5	150
5	2019-01-06	Sunday	91.0	Ada	8	0.5	120
6	2019-01-07	Monday	81.0	Ada	3	0.3	96

Python and MySQL Integration – to_csv

```
df.to_csv("cookies_clean1.csv", index= False)
```

Python Data Filtering



Python Data Filtering

```
df_filter = df['Salesman'] == 'Ada'

df[df_filter]

df_filter = (df['Salesman'] == 'Ada') & (df['Day'] == 'Monday')

df[df_filter]

df_filter = (df['Day'] == 'Tuesday') | (df['Day'] == 'Monday')

df[df_filter]

df_filter = (df['Sales'] >= 100) & (df['Tweets'] <= 1)
```

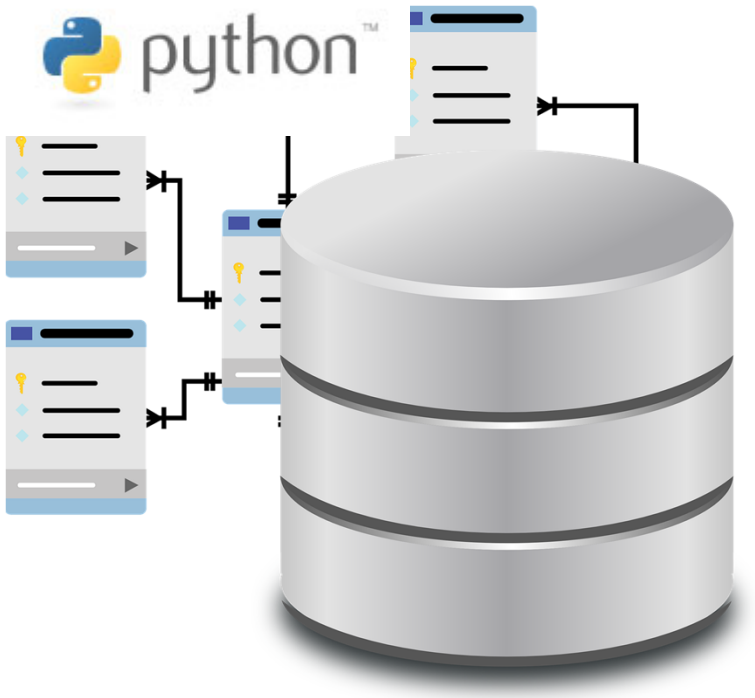
Python Data Filtering

```
df['Sales'] = pd.to_numeric(df['Sales'],errors='coerce')

df['Tweets'] = pd.to_numeric(df['Tweets'],errors='coerce')

df[df_filter]
```

Python and MySQL Integration – Pivot Table



Python and MySQL Integration – Pivot Table

```
df.pivot(index="Date", columns ="Salesman")

df.pivot(index="Date", columns ="Salesman", values = "Sales")

df.pivot(index="Salesman", columns ="Date", values = "Sales")
```

Python and MySQL Integration – Pivot Table

```
df.pivot_table(index="y", columns ="marital")

df.pivot_table(index="y", columns ="education", values = "age" )

df.pivot_table(index="marital", columns ="y", values = "balance")

df.pivot_table(index="marital", columns ="y", values = "age",
aggfunc="sum")
```